Graduate Course on Embedded Control Systems: Theory and Practice

Networks for Embedded Control Systems

Luis Almeida – University of Porto
Paulo Pedreiras – University of Aveiro

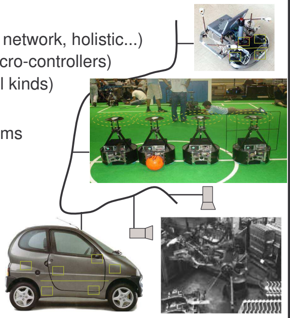
RETIS Lab
Scuola Superiore Sant'Anna, Pisa
8-12 June 2009

Outline

- ✓ **Trends in embedded systems**
  - ✓ Distribution, integration and operational flexibility
- ✓ **Timing issues in the network**
  - ✓ Timing parameters, time across the network, temporal control of the communication
- ✓ **Inside the protocol stack**
  - ✓ The physical, data link, (network) and application layers
- ✓ **A few related protocols**
  - ✓ WorldFIP, TTP/C, PROFIBUS, CAN, Ethernet, FlexRay, WiFi, ZigBee
- ✓ **Some on-going research topics**

Speakers background

- Real-time systems:
  - Scheduling aspects (processor, network, holistic...)
  - RT kernels (mainly for small micro-controllers)
  - RT communication protocols (all kinds)
- Dependable systems:
  - Architecture of embedded systems
  - Safety and reliability aspects
  - Safety-critical systems
- Main *battle fields*:
  - Dynamic reconfiguration
  - On-line QoS adaptation
  - Flexible scheduling

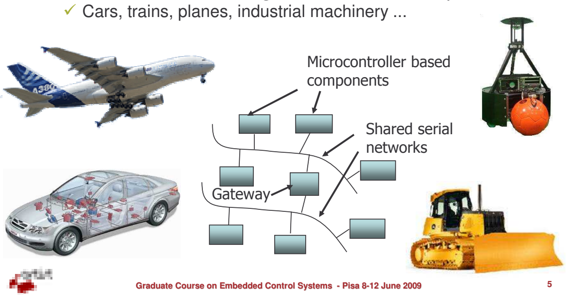


F-T-T- Flexible Time-Triggered architecture

Part 1
Trends in embedded control systems

Towards distribution

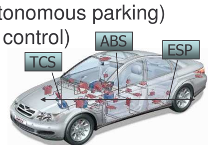
Nowadays, complex embedded systems are **distributed**, with a **network** connecting several active components
- ✓ Cars, trains, planes, industrial machinery ...



Dissemination of feedback control

The number of **automatic control loops** within complex embedded systems **increased dramatically** motivated by

- ✓Improved **performance** (e.g. TCS – Traction Control)
- ✓Increased **safety** (e.g. ABS, ESP – Elect. Stability Prog.)
- ✓Increased **energy efficiency** (e.g. Engine control)
- ✓New **functionality** (e.g. Autonomous parking)
- ✓More **comfort** (e.g. Cruise control)
- ✓**...**



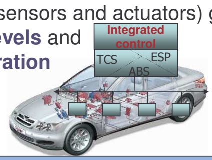
Most of such loops are distributed leading to
Distributed Computer Control Systems (DCCS)

Towards functional integration

Integrating several quasi-independent **subsystems** allows further performance improvements

- ✓ E.g. integrating ABS, ESP, TCS and Active Suspension might significantly improve car overall stability control

Using **smart components** (sensors and actuators) greatly **simplifies higher control levels** and **facilitates functional integration**

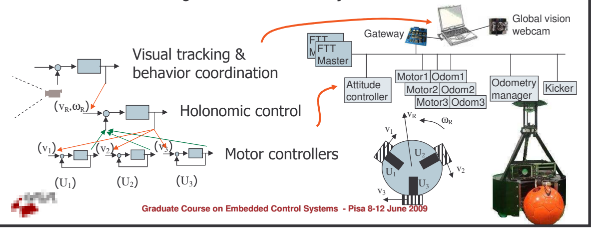


This is sometimes referred to as ***centralized* control**, but still generally using a **distributed HW architecture** !

Integration and control

***Centralized* control** is a form of **hierarchical control**

- ✓ **Integrating** the three wheels control in holonomic robots provides **full holonomic motion** (any direction with any orientation)
- ✓ Visual tracking uses **holonomic system** as a **smart actuator**

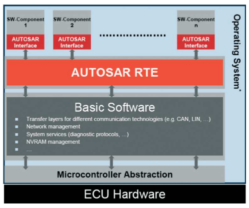


Why higher integration

- ✓ **Improve performance** by coupling subsystems (communication across functional domains)
- ✓ **Improve efficiency** in using system resources
- ✓ **Reduce** number of active **components** and **costs**
- ✓ **Manage complexity**

We need support for a **clear separation** between **SW** and **HW architectures**

- ✓ Providing SW that is unaware of the HW components
- ✓ e.g. AUTOSAR



Towards operational flexibility

Many DCCS operate in **dynamic scenarios**:

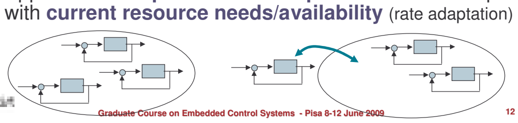
- ✓ Systems with **variable number of users** or **variable load** (traffic control, radar…)
- ✓ Systems that operate in **changing physical environments** (robots, cars…)
- ✓ Systems that can **self-reconfigure dynamically** to cope with hazardous events or evolving functionality (cars, planes, trains, production cells…)

QoS adaptation, graceful degradation, survivability

Towards operational flexibility

In other words...

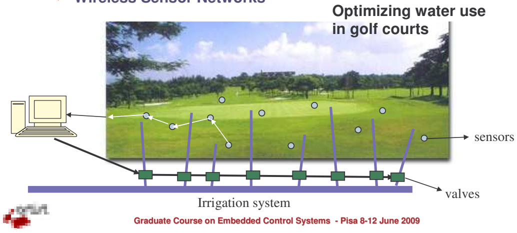
- ✓ Many **feedback control applications** will coexist and **share active resources** (ECUs and networks)
- ✓ Application **components will be added**/**replaced** during the system lifetime (system reconfiguration)
- ✓ **Backup** application components will be **switched on** at run-time to replace failed primary ones (fault tolerance)
- ✓ Application **components** will **adapt at run-time** to cope with **current resource needs/availability** (rate adaptation)



Towards sensing diffusion

- ✓ Controlling large processes
- ✓ Involving sensing in large areas
  - ✓ Precision agriculture, environment control, ...
  - ✓ **Cyber-Physical Systems**
  - ✓ **Wireless Sensor Networks**

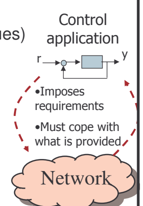
Optimizing water use in golf courts



Putting it all together

Merging **distribution, control, integration and flexibility** raises many issues that need adequate treatment

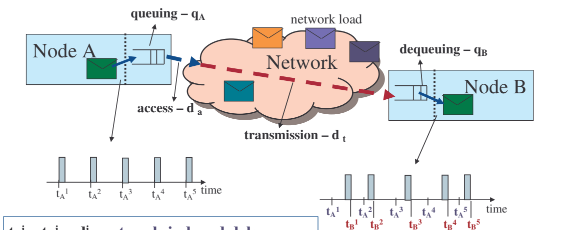
- Higher **potential for mutual interference**
  - **Higher jitter** in periodic executions
  - **Longer delays** (sampling→control→actuation)
  - **Information losses** (errors, vacant sampling issues)
  - Control **performance degradation**
  - Control **instability**...
- Conflict between **safety and flexibility**
  - Flexibility reduces **a priori knowledge**
  - Requires adequate **temporal isolation** and **temporal control**



Part 2

Timing issues in the network

Network delays



$t_B^i - t_A^i = d^i$, network-induced delay

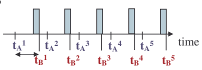
$d^i = q_A^i + d_a^i + d_t^i + q_B^i$, delay components

$d^i - d^{i-1} = j^i$, delay jitter

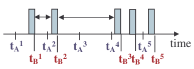
Reception instants may suffer irregular delays due to interferences from the network load, queuing policies and processor load

Delay and delay jitter

- **Network induced delay** – extra delay caused by the transmission of data over the network.
  - Some applications (e.g. **control**) are particularly sensitive to this

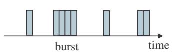


- **Delay jitter** – variation affecting the network delay.
  - Some applications (e.g. **streaming**) are not much affected by the network delay but are highly affected by delay jitter

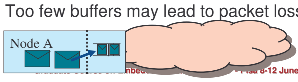


Burstiness and buffering

- **Burstiness** – measure of the load submitted to the network in a short interval of time.
  - Bursts have a profound impact on the real-time performance of the network and impose high buffering requirements.
    File transfers are a frequent cause of **bursts**.

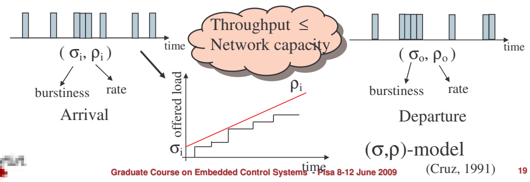


- **Buffer requirements** – Buffer capacity needed to hold packet bursts.
  - Too few buffers may lead to packet losses



Throughput and arrival / dep. rate

- **Throughput** – average amount of data, or packets, that the network dispatches per unit of time (bit/s and packets/s).
- **Arrival / departure rate** – long-term rate at which data arrives at/from the network (bit/s and packets/s).

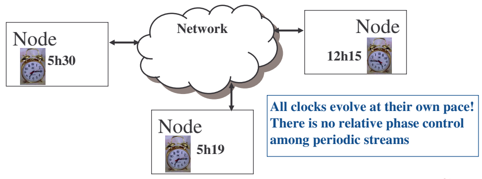


Throughput *versus* timeliness

- ***Throughput*** equates to *speed*, e.g., average number of packets delivered per unit of time
- ***Timeliness*** is related to the instant of executing certain computations (not directly related to *speed*!)
- **Timeliness** is highly influenced by the **scheduling** of activities in the system, i.e., when several packets are ready for transmission, which is sent first?

Time across a network

- ✓ As opposed to a centralized system, in a distributed system **each node has its own clock**
- ✓ Without specific support, there is no explicit coherent notion of time across a distributed systems
- ✓ Worse, due to **drift**, clocks tend to **permanently diverge**



Time across a network

- ✓ However, a **coherent notion of time** can be very important for several applications to:
  - ✓ **Carry out actions at desired time instants**
    e.g. synchronous data acquisition, synchronous actuation
  - ✓ **Time-stamp data and events**
    e.g. establish causal relationships that led to a system failure
  - ✓ **Compute the age of data**
  - ✓ **Coordinate transmissions**
    e.g. TDMA clock-based systems
  - ✓ etc.

But how to **synchronize the clocks** across the network?

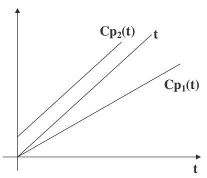
Synchronizing clocks

- ✓ Clocks can be synchronized:
  - ✓ **Externally** – an external clock source sends a time update regularly (e.g. GPS)
  - ✓ **Internally** – nodes exchange messages to come up with a global clock value
    - ✓ **Master-Slave** – A special node (time master) spreads its own clock to all other nodes
    - ✓ **Distributed** – all nodes perform a similar role and synchronize their clocks using the clocks of the others, for example, using an average (e.g. FTA, Fault-Tolerant Average)
- ✓ Standards: NTP, SNTP, IEEE 1588

Clock synchronization

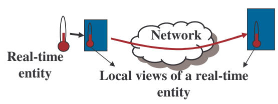
Few definitions (*$Cp_i(t)$* is the clock of node *i* at instant *t*)

- **Accuracy α**
  $|Cp_i(t) - t| \le \alpha$ for all *i* and *t*
- **Precision δ**
  $|Cp_i(t) - Cp_j(t)| \le \delta$ for all *i, j, t*
- **Offset**
  Difference between Cp(t) and t
- **Drift**
  Difference in growing rate between Cp(t) and t



Real-time messages

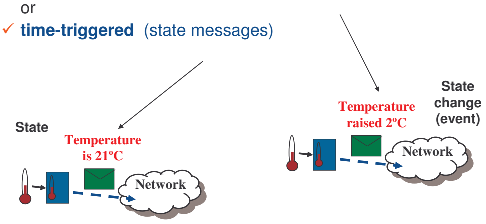
- ✓ A message related to a *real-time entity* (e.g. a sensor) is a **real-time message**.
- ✓ Real-time messages must be transmitted within precise **time-bounds** to assure **coherence** between senders and receivers concerning their **local views** of the respective real-time entities
- ✓ Real-time messages can have **event** or **state semantics**



Event or Time triggering

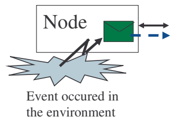
- According to the type of messages (event or state) conveyed by the network, it can be
  - **event-triggered** (event messages)

  or
  - **time-triggered** (state messages)

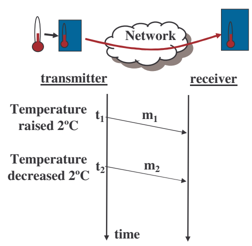


Event triggered network

- Transactions, carrying event data, are **triggered** upon **event occurrence**.
- Consequently, transmission instants are generally asynchronous wrt the nodes or network.
- Receivers know about **system state** by means of the **received events**.
- The submitted communication load (number of *simultaneous* message transmission requests) may be very high (**event showers**).



Event triggered network



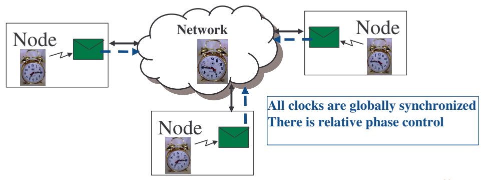
What if:
m_2 is lost?

Notice that:
$\Delta t = t_2 - t_1$ is unbounded

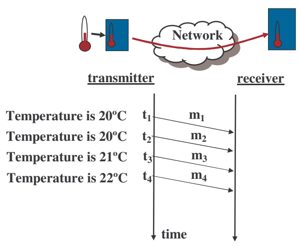
Typical solutions involve limiting Δt (e.g. with heartbeat)

Time triggered network

- There is a notion of **network time** (explicit or implicit)
- Transactions, carrying **state data**, are **triggered** at **predefined time instants**.
- Receivers have a **periodic refresh** of the system state
- The submitted communication load is well determined.



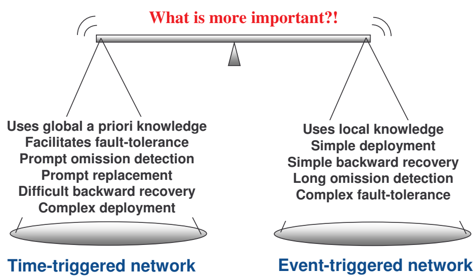
Time triggered network



Notice:
$\Delta t = t_i - t_{i-1}$ set according to system dynamics

Losing one message causes inconsistency during Δt

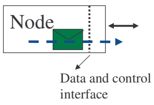
Event vs time triggering



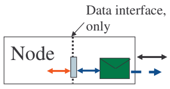
What about both?

Transmission control

- Determines **who** triggers network transactions, application or network
  - **External control**
    Transactions are triggered upon explicit control signal from the application.
    Messages are queued at the interface.
    Highly sensitive to application design/faults.

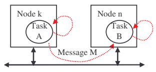


  - **Autonomous control**
    The network triggers transactions autonomously.
    No control signal crosses the CNI.
    Applications exchange data with the network by means of buffers.
    Deterministic behavior.

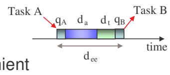


Information flow

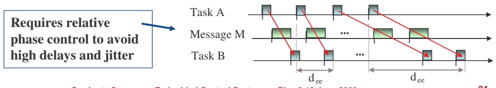
- Determining **end-to-end delay**



  - **ET-network with external control**
    Transactions are composed of several elementary actions carried out in sequence.

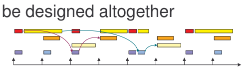


  - **TT-network with autonomous control**
    The elementary actions in each intervenient (transmitter, network, receiver) are decoupled, spinning at an appropriate rate.

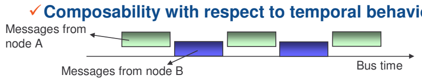


Information flow

- **In a TT-network**
  - The tight control of the relative phase required between all system activities (message transmissions and task executions) imposes **rigid architectural** constraints
    - **Time-triggered architecture**
  - The whole system must be designed altogether (network and nodes)



  - However, once the network is designed, nodes will not interfere with each other (transmissions occur in disjoint intervals)
    - **Composability with respect to temporal behavior**



Part 3
Inside the protocol stack

Interoperability

- ***Exchanging*** *information and* ***cooperating*** *with other systems (nodes)*

- Nodes in a distributed system must be interoperable

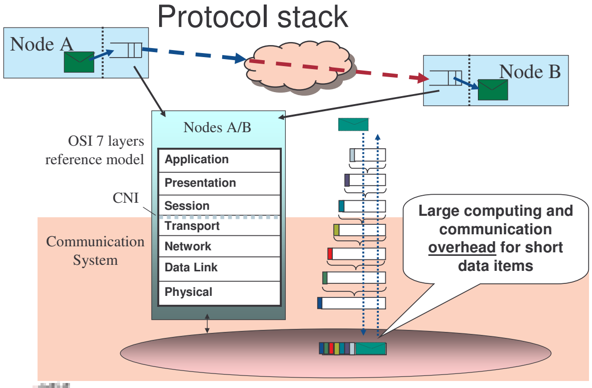
- how to assure that different nodes, with CNIs and network controllers of different makers, will be **interoperable**?

OSI reference model

- The **Open Systems Interconnection (OSI)** reference model

- Developed within **ISO** from 1977 to 1984
- Became International Standard **IS 7498**
- It is the reference model used in the development of interconnection and cooperation standards for **open distributed systems**

Protocol stack



Networks for Embedded Control Systems © Luis Almeida, Paulo Pedreiras

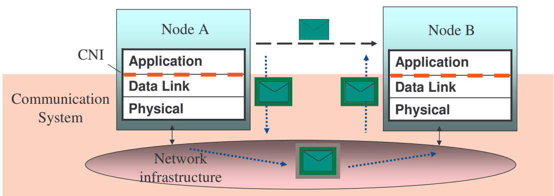
Requirements for a real-time protocol stack

- The **end-to-end communication delay** must be **bounded**
  - **All services** at all layers must be **time-bounded**
  - Requires appropriate **time-bounded protocols**
- The **7 layers** impose a considerable computation and communication **overhead**…
  - The time to execute the protocol stack may become dominant wrt the communication time
- Many real-time networks
  - are dedicated to a well defined application (no need for presentation)
  - use single broadcast domain (no need for routing)
  - use short messages (no need to fragment/reassemble)

Networks for Embedded Control Systems © Luis Almeida, Paulo Pedreiras

OSI collapsed model

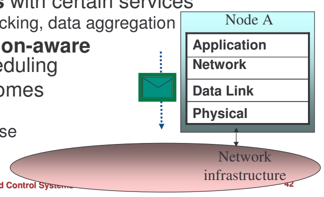
- Application services access the Data Link directly
- Other layers maybe present but not fully stacked
- In process control and factory automation these networks are called **Fieldbuses**



Networks for Embedded Control Systems © Luis Almeida, Paulo Pedreiras

Multi-hop networks

- Due to physical and/or logical constraints many control applications **cannot** be supported by ***local* networks**
  - (Wireless) sensor networks, complex industrial plants...
- A network layer with **routing** services is fundamental (multi-hop communication)
- Some more complex application layers: **Specific middlewares** with certain services
  - e.g., localization, tracking, data aggregation
- **Energy-aware** / **location-aware** routing and traffic scheduling
- **Synchronization** becomes particularly important
  - e.g., for channel reuse



Networks for Embedded Control Systems © Luis Almeida, Paulo Pedreiras

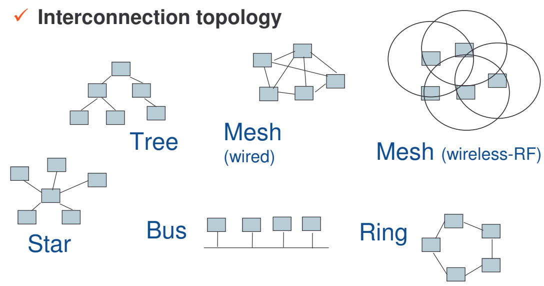
Physical layer

- <u>Issues related with the physical layer:</u>
  - **Interconnection topology**
  - **Physical medium**
  - Coding of digital information
  - **Transmission rate**
  - Maximum interconnection length
  - Max number of nodes
  - Feeding power through the network
  - **Energy Consumption**
  - Immunity to EMI
  - Intrinsic safety

Networks for Embedded Control Systems © Luis Almeida, Paulo Pedreiras

Physical layer

- **Interconnection topology**



Physical layer

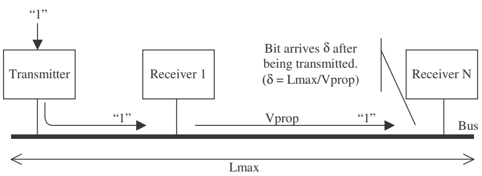
Topology	In favor	Against
Mesh	Point-to-point connections (wired, only). Several alternative paths.	Requires routing. Complex cabling (wired), difficult to maintain. Difficult to enforce total order
Tree (star)	Point-to-point connections. Simultaneous communication in parallel branches.	Requires routing. Potential long paths for deep nodes in different branches. Upper branches are bottlenecks.
Ring	Point-to-point connections. Simplified cabling.	Long path for back-to-back nodes. Depending on protocol, the whole ring is used as shared medium (more complex access control)
Bus	Simplified cabling. Direct communication (no routing)	Shared communication medium (more complex access control)

Physical layer

- ✓ **Physical medium**
  - ✓ **Copper wiring**
    - ✓ Cheaper cables and interfaces (+), suffers EMI (-)
  - ✓ **Optical fibers**
    - ✓ Immune to EMI, favors safety, wide bandwidth, low attenuation (+), expensive cables and interfaces (-)
  - ✓ **Wireless – Radio frequency**
    - ✓ Mobility, flexibility (+), very susceptible to EMI (-), multi-path fading (-), attenuation (-), open medium (+/-)
  - ✓ **Wireless – Infra-red light**
    - ✓ Mobility, flexibility, immune to EMI (+), line-of-sight (-), open medium(+/-)

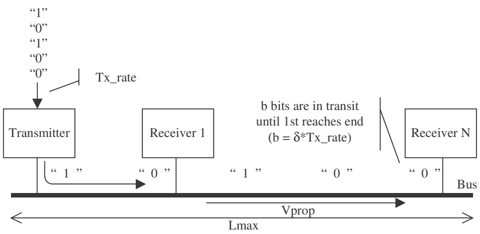
Physical layer

- ✓ **Propagation delay** in a bus
  - ✓ Time for a bit to traverse the full length of the bus (δ)



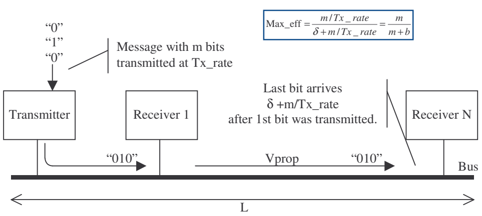
Physical layer

- ✓ **Bit length** in a bus
  - ✓ Number of bits in transit in the bus (b), given δ and Tx_rate



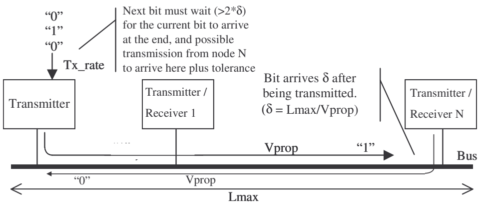
Physical layer

- ✓ **Limit to protocol efficiency in a bus**
  - ✓ Any message must be flushed from the bus before the next can be transmitted



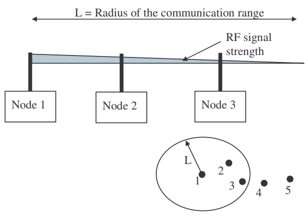
Physical layer

- ✓ **Special cases**
  - ✓ Some protocols require spatial coherence at the bit level (b=1 => Tx_rate < 1/(2δ)) (case of CAN)



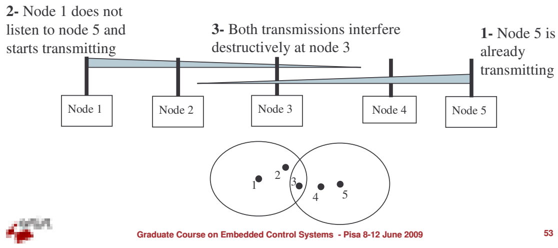
Physical layer

- **Special cases - wireless**
  - In wireless networks the **attenuation** is strong (even worse with obstacles) and transmissions from one node may not reach all nodes in the network.



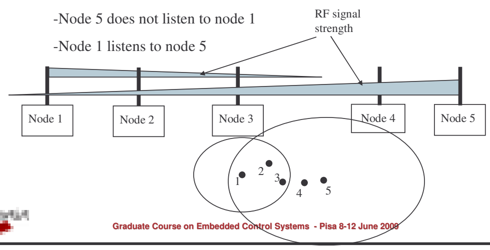
Physical layer

- **Special cases - wireless**
  - **Attenuation** is also responsible for a phenomenon called the **hidden node terminal** that jeopardizes transmission control based on carrier sensing



Physical layer

- **Special cases - wireless**
  - **Transmission power, antenna efficiency** and **local noise** deeply influences the communication range ➔ Differences cause **unidirectional** communication links



Physical layer

- **Detection of collisions**
  - In shared broadcast buses
    - If bit length **b=1**, **send 1 bit and listen**
    - If bit length **b>1**, it is not possible to relate the bit being transmitted with what is sensed on the bus. In this case a **jamming signal** is used
  - In **wireless** transmission it is not common to transmit and receive at the same time (expensive, requires multiple transceivers). **Collisions** can be sensed indirectly by means of **acknowledging**.

Physical layer

- **Energy consumption**
  - Influenced by: Transmission power, data rate, frame rate, network traffic, error control coding, retransmissions policy, type of medium access control and physical bit encoding.
  - Very **complex relationship** among the above parameters. Need to trade-off.
  - Highly relevant in **autonomous systems**, particularly in those expected to operate continuously for long term (**wireless sensor networks**).
  - **Still a nice challenge!**

Physical layer

- **Energy consumption trade-offs in wireless**
  - **Higher data rate** uses **more energy** but transmissions take **less time** (however, probability of errors and retransmissions also increases)
  - In a mesh-like network (WSNs), **higher tx power** might **reduce the number of hops**, thus less energy spent in storing and forwarding
  - **Asynchronous transmissions** (ET-like) **reduce power** on tx but receivers may need to be **awake all the time**!
  - **Collisions** require **retransmissions** but avoiding them requires a synchronization mechanism
  - …

Physical layer

- ✓ **Energy consumption in wired networks**
  - ✓ Some protocols use **transceivers** that are particularly **energy expensive** (e.g. CAN).
  - ✓ **Reducing transmissions** might be desirable in autonomous mobile robots
    - ✓ In some PIC18F458 based boards, overall current consumption raised from 70mA to 110mA corresponding to no CAN transmissions and constant transmission, respectively.
  - ✓ ...

Data link layer

- The **Data Link Layer** is concerned with local delivery of frames between devices on the same LAN.
- ✓ **Focus:**
  – Local delivery, addressing and media arbitration
- ✓ Organized in **two sublayers** (split based on the architecture of the IEEE 802 project):
  – **Logical Link Control** sublayer
    - Multiplex of high-layer protocols, flow control, error notification
  – **Media Access Control** sublayer
    - Control the access to the communication media

Data link layer

- ✓ **Addressing**
  identification of the parts involved in a network transaction
  - ✓ **Direct addressing**
    The sender and receiver(s) are explicitly identified in every transaction, using **physical** addresses (as in Ethernet)
  - ✓ **Indirect (source) addressing**
    The message **contents** are explicitly identified (e.g. temperature of sensor X). Receivers that need the message, retrieve it from the network (as in CAN and WorldFIP)
  - ✓ **Indirect (time-based) addressing**
    The message is identified by the time instant at which it is transmitted (as in TTP)

Data link layer

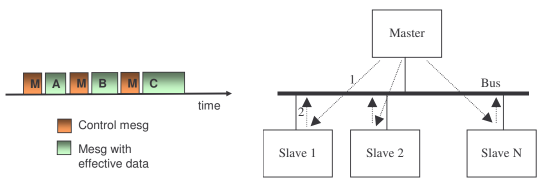
- ✓ **Logical Link Control**
  Deals with the transfer of network packets
  - ✓ Might have an impact on the network delay depending on whether **sender/receiver synchronization** is used
    - ✓ e.g., acknowledge mechanisms, retry mechanisms, request data on reply
    - ✓ Note that such synchronization using retries (PAR, ARQ) might have a strong negative impact on the data timeliness...
    - ✓ No point on continue trying to transmit out-dated values

Data Link Layer

- ✓ **Medium Access Control**
  Determines the **waiting time** to **access** the network
  - ✓ It has a large impact on the network delay
- ✓ Two main families
  - ✓ **Controlled access**
    - ✓ based on transmission control
      - ✓ The network says when to transmit
  - ✓ **Uncontrolled access**
    - ✓ based on arbitration
      - ✓ The nodes try to transmit immediately when needed

Medium Access Control
Controlled access

- ✓ **Master-slave**
  - ✓ Access granted by the Master
  - ✓ Nodes synchronized with the master
  - ✓ Requires one control message per data message

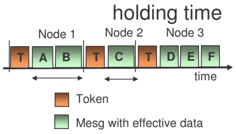


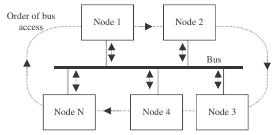
Medium Access Control
Controlled access

- ✓ **Master-slave**
  - ✓ The **traffic scheduling** problem becomes **local to the master** → good flexibility wrt scheduling algorithms
    - ✓ Requires special mechanisms to handle aperiodic communication
  - ✓ For high reliability the master must be **replicated**
  - ✓ Master messages are natural synchronization points → supports **precise tx triggering**
  - ✓ Ex: WorldFIP, Ethernet Powerlink, Bluetooth within each piconet

Medium Access Control
Controlled access

- ✓ **Token-passing**
  - ✓ Access granted by the possession of the token
  - ✓ Order of access enforced by token circulation
  - ✓ Asynchronous bus access (generally impossible to determine a priori the exact access instants)
  - ✓ Real-time operation requires bounded token holding time





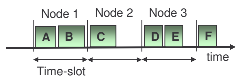
Medium Access Control
Controlled access

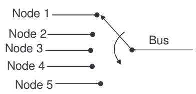
- ✓ **Token-passing**
  - ✓ **TTRT** – **Target** Token Rotation Time
    fundamental configuration parameter that determines the time the token should take in one round, under heavy traffic
  - ✓ **RTRT**- **Real** Token Rotation Time
    time effectively taken by the token in the last round
  - ✓ **ST** – Synchronous time
  - ✓ **THT** – Token **holding** time
  - ✓ High tx jitter
  - ✓ Requires mechanisms to handle **token losses**
  - ✓ Similar to **Round-Robin Scheduling**
  - ✓ Ex: IEEE 802.4, FDDI, PROFIBUS

$$\Delta = TTRT - RTRT$$
$$THT = \begin{cases} \Delta, & \Delta > ST \\ ST, & \Delta \le ST \end{cases}$$

Medium Access Control
Controlled access

- ✓ **TDMA**
  Time-Division Multiple Access
  - ✓ Access granted in dedicated time-slot
  - ✓ Time-slots are pre-defined in a cyclic framework
  - ✓ Tight synchronization with bus time (bus access instants are predetermined)
  - ✓ Requires global (clock) synchronization



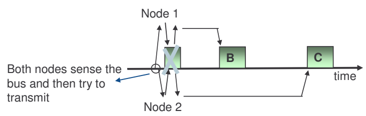


Medium Access Control
Controlled access

- ✓ **TDMA**
  Time-Division Multiple Access
  - ✓ Addressing can be based on time → **High data efficiency**
  - ✓ Quality of synchronization bounds efficiency (length of **guarding windows** between slots)
  - ✓ Typically uses **static table-based scheduling**
  - ✓ Ex: TTP/C, FlexRay-static, TT-CAN, PROFINET

Medium Access Control
Uncontrolled access

- ✓ **CSMA**
  Carrier-Sense Multiple Access
  - ✓ Set of protocols based on sensing bus inactivity before transmitting (asynchronous bus access)
  - ✓ There may be collisions
  - ✓ Upon collision, nodes back off and retry later, according to some specific rule (this rule determines, to a large extent, the real-time features of the protocol)

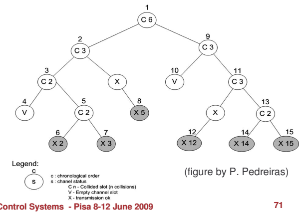


Medium Access Control
Uncontrolled access

- **CSMA/CD**
  Carrier-Sense Multiple Access with Collision Detection
  - Used in shared Ethernet (hub-based)
  - Collisions are **destructive** and are detected within **collision windows** (*slots*)
  - Upon collision, the retry interval is **random** and the randomization window is doubled for each retry until 1024 slots (BEB - Binary Exponential Back-off)
  - Non-deterministic (particularly with chained collisions)

Medium Access Control
Uncontrolled access

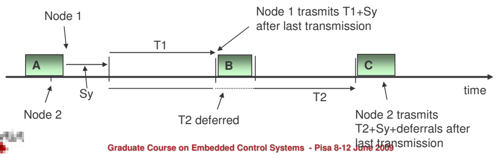
- **CSMA/DCR**
  Carrier-Sense Multiple Access with Deterministic Collision Resolution
  - Collisions are destructive
  - **back off and retry based on priority** (binary tree search)
  - Deterministic
  - Ex. G. Le Lann's modified Ethernet



(figure by P. Pedreiras)

Medium Access Control
Uncontrolled access

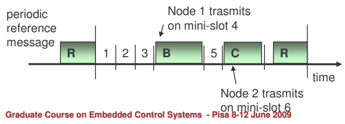
- **CSMA/CA**
  Carrier-Sense Multiple Access with Collision Avoidance (async)
  - Access based on sensing bus inactivity during a synchronization interval (Sy) plus a uniformly distributed random access interval with probability p (*p-persistent*)
  - Not collision-free (Ex: IEEE 802.11)



Medium Access Control
Uncontrolled access

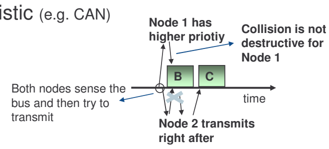
- **CSMA/CA**
  Carrier-Sense Multiple Access with Collision Avoidance (with sync)
  - Access based on sensing bus inactivity during a number of predefined time-slots (mini-slots) after reception of a synchronous reference message
  - Corresponding mini-slot determines priority
  - Collision-free and deterministic

  (Ex: FlexRay-dynamic (Byteflight), ARINC629-async)



Medium Access Control
Uncontrolled access

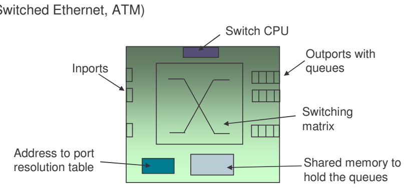
- **CSMA/BA (CA?)**
  Carrier-Sense Multiple Access with Bit-wise Arbitration
  - **Bit-wise** arbitration with **non-destructive** collisions → *Dominance protocol*
  - Upon collision, highest priority node is unaffected. Nodes with lower priorities retry right after.
  - Deterministic (e.g. CAN)



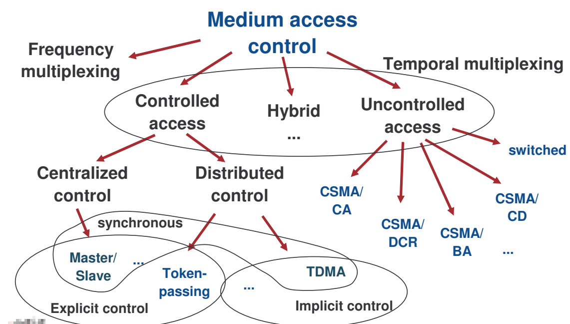
Medium Access Control
Uncontrolled access

- **Switched**
  micro-segmented network with central switching hub
  - Nodes send asynchronously messages to the switch using point-to-point links – **no collisions**
  - Contention at the network access is replaced by contention at the output ports

  (e.g. Switched Ethernet, ATM)



Data link layer



Network layer

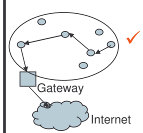
- Issues related with the network layer:
  - **Logical addressing**
  - **Routing**

Network layer

- **Logical Addressing**
  - **Independence** of higher protocols wrt the physical hardware
  - Requires an ARP (address resolution protocol)
  - Supports logical multicasts / broadcasts

Network layer

- **Routing**
  - **Ad-hoc routing:** Relaying messages between nodes within the sensor network that have no direct communication path (**communication distance larger than 1 hop**)
  - **Inter-network routing:** Relaying messages between the sensor network and the external world, e.g. a command/control system or the Internet.



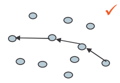
Network layer

- **Ad-hoc routing**
  - **Flat routing:**
    - All nodes have equal responsability for maintaining routes and relaying messages
  - **Hierarchical / cluster-based routing:**
    - The network is broken into geographical areas, clusters, and the inter-cluster traffic is aggregated and routed as being a single message flow.

    - Typical routing protocols include variations of the **Dijkstra's shortest path** algorithm (based on flooding) as well as of the **distance vector** technique.

Network layer

- **Routing**
  - **Proactive:**
    - Routes are maintained continuously for all reachable nodes (uses periodic route dissemination plus)
  - **Reactive:**
    - Routes are established on demand for the duration of communication sessions, only. Optimization criteria can be used to guide the route set up, e.g. minimum energy, maximum power available, maximum link strength
  - **Geographic:**
    - Uses geographical information for routing purposes. Tries to establish routes that are as close as possible to the straight line between source and sink. Leads to shorter paths and consequently to **lower end-to-end delays**.

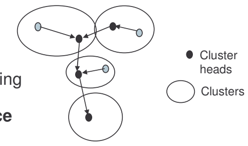


Network layer

- ✓ Basic **information propagation** techniques
- ✓ **Flooding**
  - ✓ All nodes broadcast the messages they receive until a maximum number of hops or the destination is reached
  - ✓ Simple and old technique. Presents:
    - ✓ low requirements for routing information (+)
    - ✓ too high traffic levels with many duplicates (-)
    - ✓ not resource-aware
- ✓ **Gossiping**
  - ✓ All nodes send the messages they receive to one of its neighbors randomly
    - ✓ Lower traffic levels but longer time to propagate data

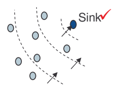
Network layer

- ✓ **Aggregation**
  - ✓ Nodes in a cluster send their inter-cluster traffic to the cluster head, that aggregates and relays it to the following cluster
  - ✓ Reduces nodes involved in routing, thus reduces also routing information
  - ✓ **Data aggregation** (combining data together) is another fundamental issue to **reduce the amount of traffic** in large sensor networks



Network layer

- ✓ **Spatial reuse**
  - ✓ Nodes that are sufficiently apart can transmit simultaneously
    - ✓ **communication range** versus **interfering range**
- ✓ **Synchronized frameworks**
  - ✓ Communication occurs in slots that are defined across the network (TDMA fashion)
    - ✓ Require synchronization (clock, beacons)
  - ✓ Enables **reducing the data routing delay** by defining adequate transmission (slot) sequences
  - ✓ Enables controlling **on-off switching for energy saving**



Protocol stack internals

- Real-time protocol stacks may be based in:
  - ✓ **Specialized communication** controllers and/or **custom device-drivers/stacks**
    - ✓ Low latency, high predictability, fine-grain control in queues, **but ...**
    - ✓ **Expensive**: non-COTS hardware, manpower for specific device-drivers development, longer development time, harder to debug, ...
  - ✓ **COTS** hardware and software
    - ✓ Cheap hardware, device drivers readily available for all the HW, full IP stacks supported, **but ...**
    - ✓ Potentially high **latency**, **unpredictability**, high **resource consumption** (memory and CPU), ...

Protocol stack internals

- Using a TCP/IP stack:
  - ✓ Easy **connection** to the **intra/Internet**, use of standard tools/apps, **easy development** (app. code independent from HW), but ...
  - ✓ "**standard**" TCP/IP stack:
    - ✓ **High CPU** and **memory** consumption (code and data memory in the order of hundreds of KB, multiple data copies); Not suitable to resource constrained embedded systems
  - ✓ "**lightweight**" TCP/IP stacks (e.g. lwIP, uIP):
    - ✓ **Code** size is around **10KB** and **RAM** size can be around **100's of B** (**suitable for 8/16 bit micro-controllers**), efficient buffer management (zero copy) , …
    - ✓ Some **limitations** (e.g. single interface, single connection)

Protocol stack internals

- Issues with general-purpose Device-drivers
  - ✓ DD **development** is **hard** and **costly**
    - ✓ Many real-time systems use general-purpose DD (GPDD), eventually with some adaptations
  - ✓ **Issues** with GPDD
    - ✓ optimized for throughput (**high latency/poor determinism**)
    - ✓ **FIFO queues** between the host memory and the NIC internal memory
    - ✓ DMA/IRQ **optimizations conflict** with **predictability**
      - – Several messages can be buffered and generate a single INT/DMA transfer; Some operations have no defined time-bound
    - ✓ E.g. RTnet (Network stack for Xenomai and RTAI) DD for 3COM NICS (rt_3c59x.c) is stated as "non real-time safe"

Application layer

- Issues related with the application layer:
- (middleware issues)
  - **Cooperation model**
    - Client-Server
    - Producer-Consumer
    - Producer-Distributor-Consumer
    - Publisher-Subscriber
  - **Examples**
    - CANOpen
    - NDDS
    - CORBA

Application layer

- Cooperation model - **Client-Server**
  - Transactions are **triggered** by the **receiver** of the requested information (**client**).
  - Nodes that **generate** information are **servers** and only react to client requests.
  - The model is based on **unicast** transmission (one sender and one receiver)
  - Transactions can be **synchronous** (client blocks until server answers) or **asynchronous** (client follows execution after issuing the request)
  - e.g. CORBA, RMI, access to SDOs in CANopen

Application layer

- Cooperation model – **Producer-Consumer**
  - Transactions are **triggered** by the nodes that **generate** information (**producers**).
  - The nodes that **need** the information, identify it when transmitted and retrieve it from the network (**consumers**)
  - The model is based on **broadcast** transmission (each message is received by all nodes)
  - e.g., PDOs distribution in CANopen, many CAN higher layers

Application layer

- Cooperation model – **Producer-Distributor-Consumer**
  - Basically similar to Producer-Consumer
  - Transactions are **triggered** by a particular node, the **distributor**, upon request from the producers or according to a pre-established schedule.
  - It is an implementation of PC over master-slave
  - e.g., WorldFIP, sync traffic in FTT-CAN

Application layer

- Cooperation model – **Publisher-Subscriber**
  - Elaborate version of Producer-Consumer using the concept of **group communication**
  - Nodes must adhere to groups either as publisher (produces information) or as subscriber (consumes information)
  - Transactions are **triggered** by the **publisher** of a group and **disseminated** among the respective **subscribers**, only (**multicast**).
  - e.g., DDS, sync traffic in FTT-SE

Application layer

- Some Application Layer examples
  - CANOpen
  - NDDS
  - CORBA

CANopen

- CANopen
  - communication and application standard for distributed systems
  - Maintained by the CAN-in-Automation (CiA) group http://www.can-cia.com
  - Key features:
    - Transmission process data according to the producer/consumer model
    - Standardized device description (data, parameters, functions, programs)
    - Standardized access to device parameters
    - Standardized services for device monitoring (e.g. membership functions based e.g. in heartbeat)
    - System services: synchronization message, central time-stamp message (e.g. synchronous data acquisition)
    - Emergency messages
    - Adopted by other protocols (e.g. Ethernet POWERLINK)

CANopen protocols

- **Process Data Objects** (PDO)
  - ✓ Carry actual application data; broadcast, producer/consumer cooperation model, **unacknowledged**

□ PDO types
  - ✓ ***Asynchronous PDOs***
    event-controlled, automatically transmitted when at least one of the process variables mapped in a PDO is altered
  - ✓ ***Synchronous PDOs***
    transmitted after reception of synchronization message (Sync Object).
    - Transmission is carried out synchronously in the entire network.
    - All device inputs are sampled on the arrival of the sync object
    - Sampled data transmitted after the next sync message (1 cycle constant delay)

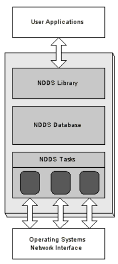
CANopen

•**Device configuration (service data objects/ SDO)**
- ✓ SDOs are used to read/write device OD entries
- ✓ Client/server, logical 1:1 channel
- ✓ Acknowledged service: each client SDO requires a server answer

淵**Network management (NMT)**
- ✓ Network management services for **control** of the **communication state** of network nodes and **node monitoring**
  - Node states: Initialization / Pre-operational (used for configuration, no PDO transactions) / Operational / Stopped (only heratbeat/guard messages can be txmitted)
  - Network master may change the node's state

DDS

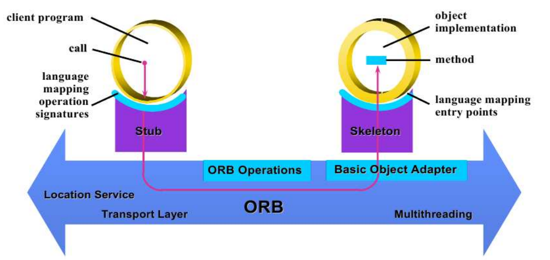
- **Data Delivery Service** (DDS)
- ✓ Middleware for **distributed real-time applications** made by Real-Time Innovation, Inc.
- ✓ Based on the Real-Time **Publisher-Subscriber** model
- ✓ Standardization within the Object Management Group (OMG)
- ✓ DDS database **shared** among **all nodes**, which have an holistic view on the communication requirements
- ✓ Publishers **create** "**topics**" (e.g. temperature) and **publish "samples"**
- ✓ The **network delivers** the "samples" to each subscriber
- ✓ Addressing, delivery, flow control, etc. handled by the DDS



CORBA

- **C**ommon **O**bject **R**equest **B**roker **A**rchitecture
- ✓ **Open specification** proposed by the OMG http://www.omg.org
- ✓ Objective: Clients use **remote objects as if they were local**
- ✓ Characteristics:
  - ✓ **Multiplatform**: Windows, Linux, Unix, MacOS, …
  - ✓ **Multilingual**: Ada, C, C++, Java, Python, …
  - ✓ **Interoperability** between lenguages and platforms
  - ✓ **Multiple vendors** (some are freeware products)

CORBA



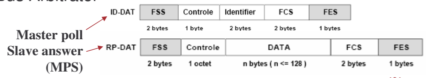
Isidro Calvo, Dpto. Ingenieria de Sistemas y Automatica, Escuela Superior de Ingenieria de Bilbao

Part 4
Some related network protocols

WorldFIP
Factory Instrumentation Protocol

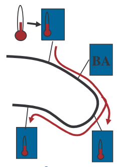
www.worldfip.org

- Created, in the 80s, in France for use in **process control** and **factory automation**.
- 2 messaging systems:
  - **MPS – real-time** services, periodic, aperiodic;
    MMS (ISO 9506) **subset** – non-real-time messaging
- Data **payload** between **0 and 128 bytes** (256 for NRT)
- **Source-addressing** (message identifiers with 16 bits)
- **Master-Slave** bus access control
  - BA - Bus Arbitrator



WorldFIP

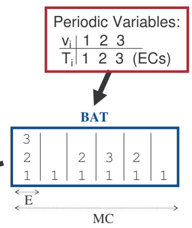
- **MPS** – *Messagerie Periodique e Sporadique*
- **Producer-Distributor-Consumer** model
- Concept of **Network Variable**
  - Entity that is distributed (several local copies coexist in different nodes)
  - Can be **periodic** or **aperiodic**
  - Local copies of **periodic variables** are **automatically refreshed** by the network
  - Local copies of **aperiodic variables** are refreshed by the network upon **explicit request**

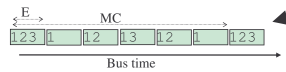


WorldFIP

- **Table-based** scheduling of **periodic** traffic
- Table (BAT) organized in **cycles**
  - **Elementary cycles**
    duration E (=GCD of periods)
    Macro-cycles
    LCM of periods (in ECs)
- Scheduling model

$\Gamma_p \equiv \{v_i : v_i(C_i, T_i, D_i, O_i), i=1..N_p\}$
$i=1..N_p$, $C_i << E$, T_i, O_i integer mult. of E

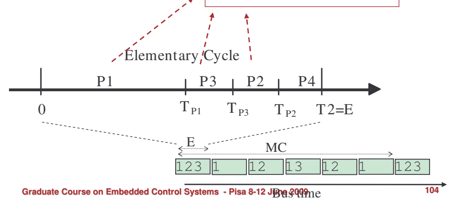




WorldFIP

- Elementary cycles organized in **phases**:
  - **Periodic** (P1)
  - **Aperiodic** (P3)
  - MMS **messages** (P2)
  - Sync - **padding** (P4)

Each **phase** is a **server** that can schedule a class of traffic



TTP/C
Time-Triggered Protocol

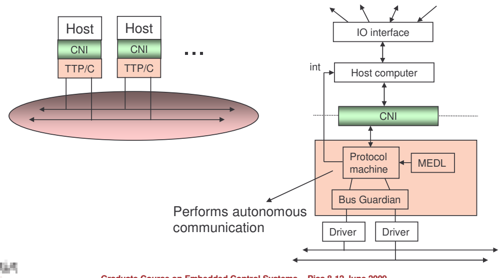
(TTP/C – for SAE class C)

www.tttech.com

- Created around 1990 within the **MARS project** in the Technical University of Vienna
- Aims at **safety-critical** applications
- Considers an architecture with nodes integrated in **fault-tolerant units** (FTUs), interconnected by a **replicated bus**
- Includes support for prompt **error detection** and **consistency checks** as well as **membership** and **clock synchronization** services.

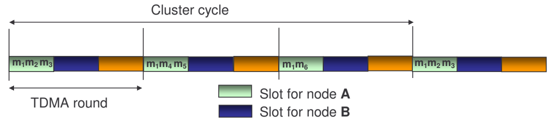
TTP/C

✓ **Network and nodes architecture**



TTP/C

- ✓ **TDMA** access with **one slot** allocated **per node** and **per round**
- ✓ The periodic sequence of slots is a **TDMA round** (typical values 1-10ms)
- ✓ In **each slot** the respective node may send **one frame** (up to 240B)
- ✓ Each frame may contain **several messages**
- ✓ The periodic sequence of messages is a **Cluster cycle**
- ✓ All message transmission instants are stored in a **distributed static table**, the **MEDL**



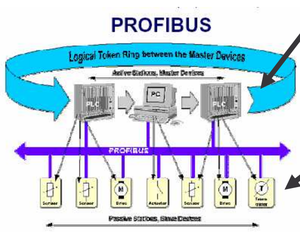
PROFIBUS
PROcess FieldBUS

www.profibus.com

- Created in the late 80's by **Siemens,** in Germany
- Two main application profiles:
  - **PROFIBUS** / **FMS** - Fieldbus Message Specification
  - **PROFIBUS** / **DP** - Decentralised Peripherals
    - The most common profile (~90%)
- Data **payload** between **0** and **246 bytes**
- **Direct-addressing** (1 byte, possibly extended)
- **Hybrid** bus access control
  - **Token-passing** among masters
  - **Master-Slave** in each individual data transaction

PROFIBUS

- **General architecture**

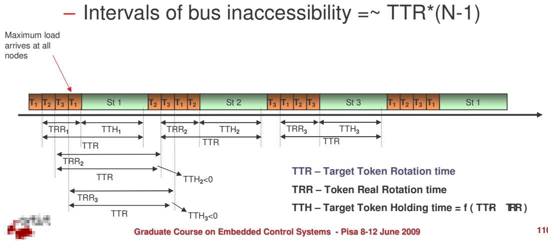


Masters: controllers, regulators, … (PC, PLC,…)

Slaves: sensors, actuators (motors, valves, thermometers…)

PROFIBUS

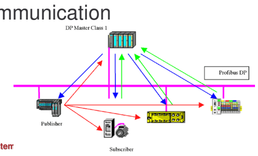
- **Traffic pattern under maximum load:**
  - Equal distribution of bandwidth among N nodes
  - Intervals of bus inaccessibility =~ TTR*(N-1)



TTR – Target Token Rotation time
TRR – Token Real Rotation time
TTH – Target Token Holding time = f (TTR TRR)

PROFIBUS

- Recent support for: (PROFIBUS / DP-v2)
  - **Isochronous** traffic
    - **Equidistant** token arrivals forming **cycles**
    - Cycle synchronization (specific global frame - broadcast)
    - Two windows in the cycle (isochronous and async traffic)
  - **Direct slave to slave** communication
    - **Producer – Consumer** style
  - Specifically developed for high accuracy drives

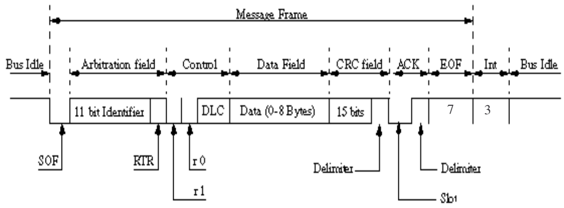


CAN
Controller Area Network
www.can-cia.de

- Created in the early 90s by **Bosch**, GmbH, for use in the **automotive industry**.
- Data **payload** between **0 and 8 bytes**
- **Source-addressing** (message identifiers) with 11 bits in version A and 29 bits in version B
- **Asynchronous** bus **access**
- **Non-destructive arbitration** based on message identifiers (establish priority)
  - Bit-wise deterministic collision resolution **CSMA/BA** (CA?,DCR?)

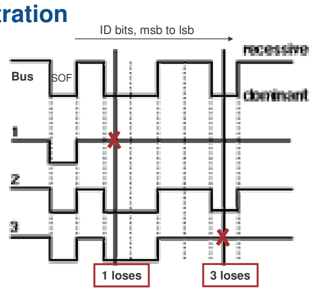
CAN

- **CAN 2.0A message frame**



CAN

- **Prioritized arbitration**
  - All nodes transmit and listen every bit
  - If different, then lost arbitration and backoff



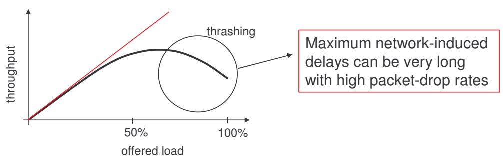
Ethernet
standards.ieee.org/getieee802/802.3.html

- **Created in the mid 70s** (!) by Robert Metcalfe at the Xerox Palo Alto Research Center.
- **CSMA/CD** non-deterministic arbitration (used in shared Ethernet, only)
- **1-persistent** transmission (transmits with 100% probability as soon as the medium is considered free)
- Collisions can occur during the interval of one ***slot*** after start of transmission (512 bits)
- When a collision is detected a **jamming** signal is sent (32 bits long)
- Frames vary between 64 (min) and 1518 (max) octets

Ethernet

- **Thrashing effect**

  As the offered load increases,
  the actual load handled by the network (throughput) decreases



Ethernet

- **Motivations for using Ethernet**
  in application domains beyond its original one
  e.g. factory automation, large embedded systems
  (Decotignie, 2001)
  - Cheap wrt other high speed technologies
  - Widely available
  - Scalable tx rate
  - High bandwidth
  - Easy integration with office networks (important for logging, management and multi-level integration, e.g. CIM)
  - IP stacks widely available
  - Well known and mature technology

Ethernet

- **Downside of using Ethernet**
  in application domains beyond its original one
  e.g. factory automation, large embedded systems
  (Decotignie, 2001)
  - Connection costs higher than traditional fieldbuses (PHY, transformer, MAC)
  - High communication overhead (for short data items)
  - High computing power requirements (for efficient bandwidth usage)
  - Typical star topology not always adequate (may lead to extra long cabling)
  - Existing protocol stacks (mainly IP) not optimized for real-time operation

Ethernet

- **Making Ethernet real-time**
  - Keep **network load low** (~1%, e.g. NDDS) !
  - **Avoid bursts** (e.g. traffic smoothing)
  - **Modify the back-off and retry** mechanism (e.g. CSMA/DCR, Virtual-time, windows, EQuB)
  - **Control transmission** instants (e.g. token-passing (RETHER, RT-EP), master-slave (FTT-Ethernet, ETHERNET-Powerlink), TDMA (PROFINET-IRT),…)
  - Micro-segment the network using **switches** (IEEE 802.1D) – Avoids collisions!

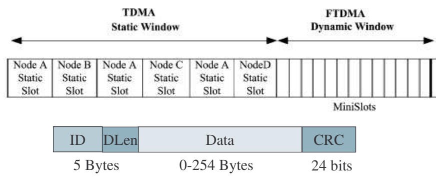
Ethernet

- **Using switches**
  - Became the **most common** solution
    - Current switches are *wire-speed* (non-blocking)
    - 802.1D – possibly with multiple priority queues (802.1p)
    - 802.1Q – Virtual LANs
  - **Not perfect !**
    - **Priority inversions** in queues (normally FIFO)
    - **Mutual interference** through shared memory and CPU
    - **Additional forwarding delay** (with jitter caused by address table look up, address learning, flooding)
    - **Delays vary** with switch technology and internal traffic handling algorithms

FlexRay

www.flexray.com

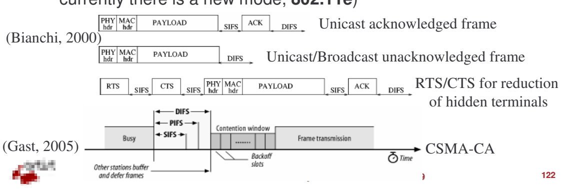
- **Created in the early 2000s** by an industrial consortium from the automotive domain.
- First public specifications available since 2004 (several changes before that)
- Is expected to replace CAN and contend with TTP/C !
- Tx rate up to 10Mbit/s



WiFi (802.11)

standards.ieee.org/getieee802/802.11.html

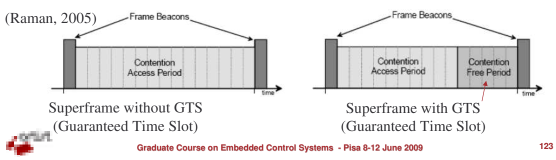
- **Created in the 1990s** for the SOHO domain. As Ethernet, 802.11 became **widely used** as **wireless** communication **infrastructure**.
- 3 modes/bands: 802.11b/g (ISM-2.4GHz), 802.11a (5GHz)
- Scalable Tx rates between 1Mbit/s and 54Mbit/s
- Many mechanisms to **reduce collisions** and **hidden-terminals**
- Growing interest for **QoS support** (initial PCF mode was abandoned, currently there is a new mode, **802.11e**)



ZigBee

www.ZigBee.org

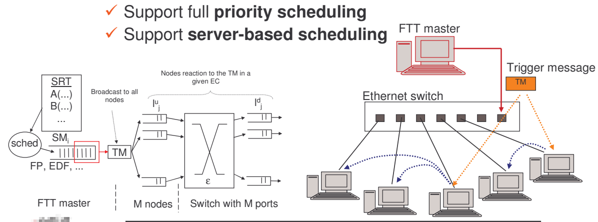
- **Created in the early 2000s** for the **wireless monitoring and control**. Targets **wireless sensor networks.**
- Works on top of 802.15.4, a PAN DLL. 3 bands: ISM-2.4GHz, 868/915 MHz. Targets very low consumption.
- Data rate of **250 Kbit/s** with range up to 300m
- **Beacon mode** and possible **PAN coordinator** to structure cells and QoS support. **Structured** and **peer-to-peer** modes. Routing support.
- Up to **65K nodes** (full and reduced function devices)



Part 5
Some related on-going research

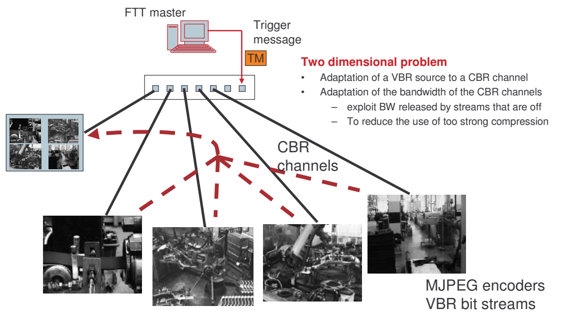
FTT-SE

- **Keeping under control** the traffic **load** submitted to a switched network
  - Schedule traffic per cycles
  - Submit only the traffic that fit in a cycle
  - **Eliminate memory overloads**
  - Support full **priority scheduling**
  - Support **server-based scheduling**



R. Marau, L. Almeida, P. Pedreiras. Enhancing Real-Time Communication over COTS Ethernet switches. WFCS 2006, IEEE 6th Workshop on Factory Communication Systems, Turin, Italy. June 2006.

QoS adaptation with FTT-SE



J. Silvestre, L. Almeida, R. Marau, P. Pedreiras. Dynamic QoS Management for Multimedia Real-Time Transmission in Industrial Environments. ETFA 2007, Patras, Greece, September 2007.

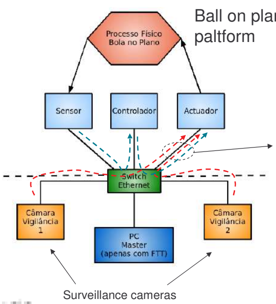
5 Mbit/s
V
1 Mbit/s

after 20s

1 Mbit/s
V
5 Mbit/s

6Mbit/s at all times

Server-SE



Ball on plane paltform

Experiments with FTT-SE / Server-SE and plain SE

(see videos)

Total load varied between 18Mbit/s and 91Mbit/s

1 short control packet together with 2 long video frames

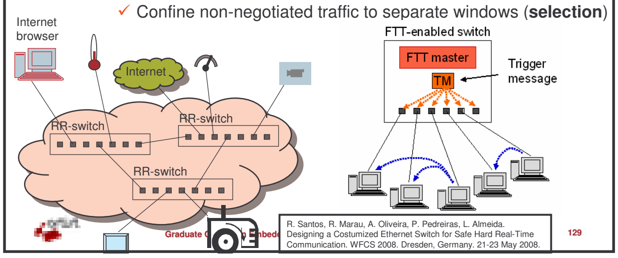
Without isolation, too high load causes control packet losses and large jitter

Surveillance cameras

Nuno Figueiredo. Controlo Distribuído de Plataformas para Experiências de Mecatrónica. Master Dissertation. University of Aveiro, July 2008. (in Portuguese)

Resource-reservation switch

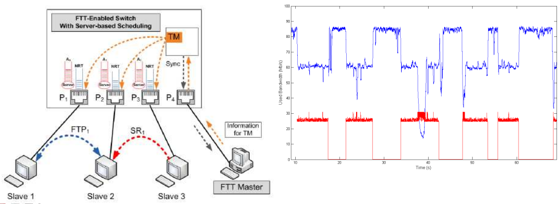
- Providing timeliness, flexibility and high robustness in switched Ethernet networks
  - Enforce negotiated channel characteristics (**policing**)
  - Reject abusive negotiated traffic (**filtering**)
  - Confine non-negotiated traffic to separate windows (**selection**)



R. Santos, R. Marau, A. Oliveira, P. Pedreiras, L. Almeida. Designing a Costumized Ethernet Switch for Safe Hard Real-Time Communication. WFCS 2008. Dresden, Germany. 21-23 May 2008.

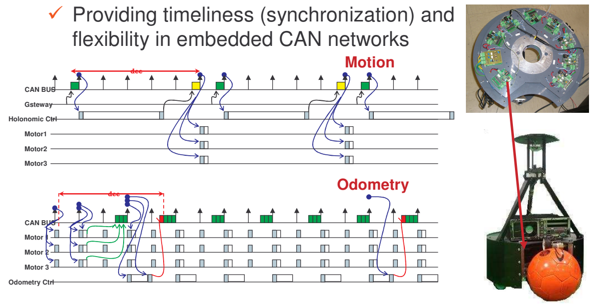
Resource-reservation switch

- 1 **real-time aperiodic** stream (switched on and off)
- 1 **non-real-time stream** (bursty and persistent)
  - **Large file transfer**
- Sent to the same output link



FTT-CAN

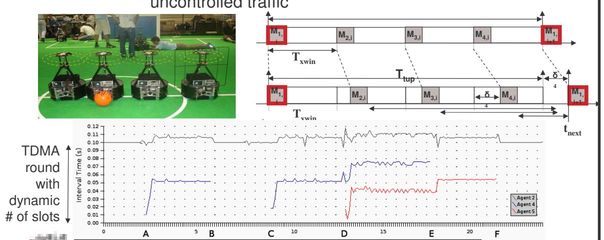
- Providing timeliness (synchronization) and flexibility in embedded CAN networks



V. Silva, R. Marau, L. Almeida, J. Ferreira, M. Calha, P. Pedreiras, J. Fonseca. Implementing a distributed sensing and actuation system: The CAMBADA robots case study. ETFA 2005, Catania Italy, September 2005.

Robust IEEE802.11 communication

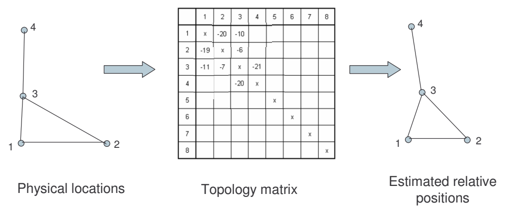
- Providing robust IEEE 802.11 communication for teams of autonomous mobile robots
  - Reconfigurable and adaptive TDMA to cope with uncontrolled traffic



F. Santos, L. Almeida, L. S. Lopes. Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots. ETFA 2008, Hamburg, Germany, 15-18 September 2008.

Relative localization with IEEE802.15.4

- Using IEEE 802.15.4 communication to
  - **Discover and track the topology** of a team of autonomous mobile robots
  - **Find their relative positions** for team coordination



Hongbin Li, Luis Almeida, Zhi Wang, Youxian Sun. Relative Positions within Small Teams of Mobile Units. MSN 2007, 3rd Int. Conf. on Mobile Ad-hoc and Sensor Networks. Beijing, China. Dec 2007.

Conclusion

- The **network** is a **fundamental** component within a distributed system (supports system integration)
- **Real-time operation** requires **time-bounded** communication
  - appropriate **protocols** must be used
- **Growing distribution, integration** and **flexibility** in distributed embedded systems require more support from the network to **maintain temporal guarantees, safety** and **robustness**
- Still many open issues in trying to improve the **timeliness of the communication**, mainly in multi-hop or multi-segment scenarios together with **reconfigurability and adaptivity, and high dependability** (reliability, availability, security, …)

Bibliography

- N. Navet, F. Simonot-Lion (eds.). ***Automotive Embedded Systems Handbook***. CRC Press, 2008.
- Richard Zurawski (ed.). ***The Industrial Communication Systems Handbook***. CRC Press, 2005.
- B. Bouyssounouse, J. Sifakis (eds.) ***Embedded Systems Design, The ARTIST Roadmap for Research and Development.*** LNCS 3436, Springer 2005.
- Le Boudec, J.-Y., Thiran, P., **Network Calculus: a theory of deterministic queuing systems for the Internet**. Springer-Verlag, Vol. 2050, 2001 (available for free download).
- P. Veríssimo, L. Rodrigues. ***Distributed Systems for System Architects***. Kluwer Academic Publishers, 2001.
- J. Liu. ***Real-Time Systems***. Prentice-Hall, 2000.
- Krishna and Shin. ***Real-Time Systems***. McGraw Hill, 1997.
- Kopetz H.. ***Real-Time Systems: Design Principles for Distributed Embedded Applications***. Kluwer Academic Publishers, 1997.

Other suggested reading

- J.-D. Decotignie. **Ethernet-based Real-time and Industrial Communications**. In Proceedings of the IEEE, volume 93, pages 1102–1117, June 2005.
- O. Redell, J. Elkhoury, M. Törngren. **The AIDA tool-set for design and implementation analysis of distributed real-time control systems**. Microprocessors and Microsystems, 28(4):163-182, May 2004.
- J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou. **Realtime communication and coordination in embedded sensor networks**. In Proceedings of the IEEE, volume 91, pages 1002–1022, July 2003.
- P. Koopman. **Critical Embedded Automotive Networks.** *IEEE Micro*, IEEE Press, July/August 2002.
- J.-D. Decotignie. **Wireless fieldbusses – a survey of issues and solutions**. In Proc. 15th IFAC World Congress on Automatic Control (IFAC 2002), Barcelona, Spain, July 2002.
- Thomesse J.-P.. **A Review of the Fieldbuses.** *Annual Reviews in Control*, 22:35-45, 1998.
- M. Törngren. **Fundamentals of implementing Real-time Control applications in Distributed Computer Systems**. *Journal of Real-time Systems,* 14:219-250. Kluwer Academic Publishers, 1998.
- Malcolm N. and W. Zhao. **Hard Real-Time Communication in Multiple-Access Networks.** *Journal of Real-Time Systems*, 8(1): 35-78, 1995.
- Tindell K., A. Burns and J. Wellings. **Analysis of Hard Real-Time Communication.** *The Journal of Real-Time Systems*. 9:147-171, Kluwer Academic Press. 1995.

Announcement

A simulation competition, you just have to write the robots program!

- **CyberRescue@RTSS2009**
  - **http://robot.unipv.it/cyberrescue-RTSS09**
- *Control the team of 5 robots with **ad-hoc communication capabilities** to reach the victim in the least time*